UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1156

Programme 3
Réseaux et Systèmes Répartis

ON THE OPTIMAL STOCHASTIC SCHEDULING OF OUT-FORESTS

Edward G. COFFMAN, Jr.
Zhen LIU

Février 1990

RR_1156

Sur l'Ordonnancement Optimal des Forêts Divergentes

Edward G. COFFMAN, Jr.

AT & T Bell Laboratories
Murray Hill
New Jersey 07974
U.S.A.

Zhen LIU

INRIA
Centre Sophia Antipolis
06565 Valbonne Cedex
France

Résumé

Cet article présente de nouveaux résultats d'ordonnancement optimal de tâches sur $K \geq 1$ processeurs parallèles, où l'objectif est de minimiser le temps total d'achèvement des tâches pour l'ordre stochastique fort. Les tâches satisfont des contraintes de précédence de type forêt divergente, c'est-à-dire que chaque tâche a au plus un prédécesseur immédiat. Les temps d'exécution des tâches sont des échantillons d'une distribution exponentielle donnée. Nous définissons la classe des forêts divergentes *uniformes* où les sous-arbres sont ordonnés par une relation d'inclusion. On prouve qu'une politique intuitive gourmande est optimale pour $K = 2$, et que si les forêts divergentes satisfont une contrainte supplémentaire d'inclusion par racine, alors la politique gourmande est optimale pour tout $K \geq 2$.

On the Optimal Stochastic Scheduling of Out-Forests

Edward G. Coffman, Jr.

AT&T Bell Laboratories
Murray Hill, New Jersey 07974
USA

Zhen Liu

INRIA
Centre Sophia Antipolis
06565 Valbonne Cedex
FRANCE

ABSTRACT

This paper presents new results on the problem of scheduling jobs on $K \geq 1$ parallel processors so as to minimize stochastically the makespan. The jobs are subject to out-forest precedence constraints, i.e. each job has at most one immediate predecessor, and job running times are independent samples from a given exponential distribution. We define a class of *uniform* out-forests in which all subtrees are ordered by an embedding relation. We prove that an intuitive greedy policy is optimal for $K=2$, and that if out-forests satisfy an additional, uniform *root-embedding* constraint, then the greedy policy is optimal for all $K \geq 2$.

On the Optimal Stochastic Scheduling of Out-Forests

Edward G. Coffman, Jr.

AT&T Bell Laboratories
Murray Hill, New Jersey 07974
USA

Zhen Liu

INRIA
Centre Sophia Antipolis
06565 Valbonne Cedex
FRANCE

1. Introduction

This paper studies the problem of scheduling $N \geq 1$ stochastic jobs on $K \geq 1$ identical processors so as to minimize stochastically the makespan. Job running times are independent samples from a given exponential distribution, and the precedence constraints among jobs form an out-forest (see Fig. 1 for an illustration). Thus, every job has at most one immediate predecessor; a job without any predecessors forms the root of one of the out-trees of the out-forest.

Reversing out-forest precedence constraints (reversing the arrows in Fig. 1) produces an in-forest. Stochastic scheduling of in-forests has been the subject of a considerable body of research. Chandy and Reynolds [3] proved that, when $K=2$, the preemptive (respectively non-preemptive) Highest-Level-First (HLF) rule minimizes expected makespan among the class of preemptive (respectively non-preemptive) policies. Here, the *level* of a job is simply the distance from it to the root of the tree in which it appears. Bruno [2] subsequently showed that HLF *stochastically* minimizes the makespan. Pinedo and Weiss [6] obtained similar results for the case where jobs at different levels are allowed to have different expected running times. Frostig [4] later extended the model in [6] to include families of increasing-likelihood-ratio distributions. Papadimitriou and Tsitsiklis [5] showed that in the Chandy-Reynolds model with K arbitrary HLF is asymptotically optimal as $N \to \infty$. Baccelli and Walrand [1] studied the problem in a fully

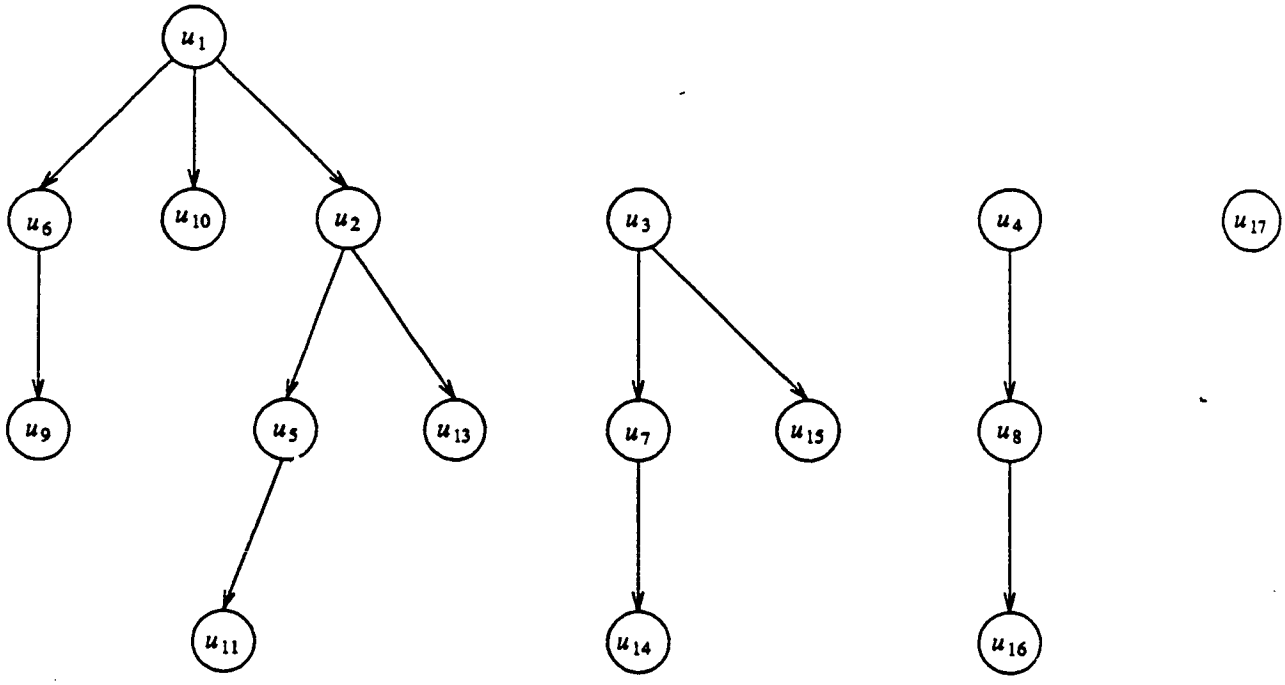


Figure 1 – An Out-Forest of 4 Out-Trees, $T(u_1)$, $T(u_3)$, $T(u_4)$, $T(u_{17})$. Jobs are indexed in largest-subtree-first order.

stochastic setting, where in-forests arrive in a Poisson stream. They showed that for $K=2$ HLF minimizes the expected number of forests in system.

Of course, there has also been a longstanding interest in the corresponding out-forest problems. But these problems appear to be much more difficult. This fact is obscured by the simple relation between in-forests and out-forests, and may be surprising at first glance. An aim of this paper is further insight into this greater difficulty. To this end, we shall define a class of *uniform* out-forests and show that for $K=2$ the greedy policy analogous to HLF stochastically minimizes makespans for such out-forests. By means of counterexamples, we illustrate how greedy policies fail to be optimal when attempts are made to extend significantly the class of uniform out-forests. Finally, for K arbitrary, we exhibit a subset of the uniform out-forests for which the greedy rule is optimal.

2. Preliminaries

Definitions – We define an out-forest F as a directed acyclic graph on a set of jobs $\{u_1, \ldots, u_N\}$. Then the set of directed edges in F is such that for all u_i in F, u_i has at most one immediate predecessor. F is an out-tree if it has exactly one job (root) with no predecessors. Hereafter, the terms forest and tree will refer to out-forest and out-tree, respectively. Also, in a harmless, but convenient abuse of notation, we shall often use F to denote just the set of jobs, as in the expression $u_i \in F$.

The job $u_i \in F$ and all of its successors is a subtree denoted by $T(u_i)$. The *depth* $d(u_i)$ of u_i is the maximum of the lengths of the paths from u_i to the leaves of $T(u_i)$. Thus, if $s(u_i)$ denotes the set of immediate successors of u_i, then $d(u_i) = 1$ if u_i is a leaf and $d(u_i) = 1 + \max_{v \in s(u_i)} d(v)$ otherwise. The number of jobs, or *size* of $T(u_i)$, is denoted by $z(u_i)$. If u_i is not a root of F, then $p(u_i)$ denotes the immediate predecessor of $u_i \in F$.

A tree T_1 is said to *embed* the tree T_2, or T_2 *is embedded in* T_1, if the 'pattern' represented by T_2 exists in T_1. We write $T_1 \geq_e T_2$ or $T_2 \leq_e T_1$. Formally, let $s_1(u)$, respectively $s_2(u)$, denote the set of immediate successors of u in T_1, respectively T_2. Then T_1 embeds T_2 if there exists a one-to-one mapping f from the jobs of T_2 into the jobs of T_1 such that for all $u, v \in T_2$, $v \in s_2(u)$ implies $f(v) \in s_1(f(u))$. We write $f(T_2)$ to denote the image of T_2 in T_1, and we call f an *embedding function*.

A forest F with jobs $u_1, \ldots, u_N$ is said to be *uniform* if the set of all its subtrees $\{T(u_i),\ 1 \leq i \leq N\}$ can be ordered by the embedding relation; by convention the indexing of jobs is assumed to be such that $T(u_1) \geq_e T(u_2) \geq_e \cdots \geq_e T(u_N)$ is such an ordering. Figure 1 shows a uniform forest. We emphasize the recursive nature of the definition; just a collection of trees ordered by the embedding relation is not necessarily a uniform forest (see Fig. 2). The embedding ordering must extend to the set of all subtrees of the forest.

The embedding relation is extended to uniform forests as follows. Let the jobs of the uniform forests F_1 and F_2 be $u_1, \ldots, u_{N_1}$ and $v_1, \ldots, v_{N_2}$, respectively, so that by convention, $T(u_i) \geq_e T(u_{i+1})$, $1 \leq i < N_1$, and $T(v_i) \geq_e T(v_{i+1})$, $1 \leq i < N_2$. Then F_1 *embeds* F_2, or F_2 *is embedded in* F_1, if $N_2 \leq N_1$ and $T(v_i) \leq_e T(u_i)$, $1 \leq i \leq N_2$; we write $F_1 \geq_e F_2$ or $F_2 \leq_e F_1$.

To each job u_i in a given forest we associate a random variable τ_i denoting the running time of u_i on any of the K identical processors $P_1, \ldots, P_K$. The τ_i's are independent samples from a given exponential distribution. Except for numerical work, the parameter of the distribution is left unspecified, as it will play no role in characterizing optimal scheduling policies.

Scheduling Policies – Let π denote an arbitrary scheduling policy. With K given, the makespan (latest job finishing time) of the schedule produced by π for forest F on K

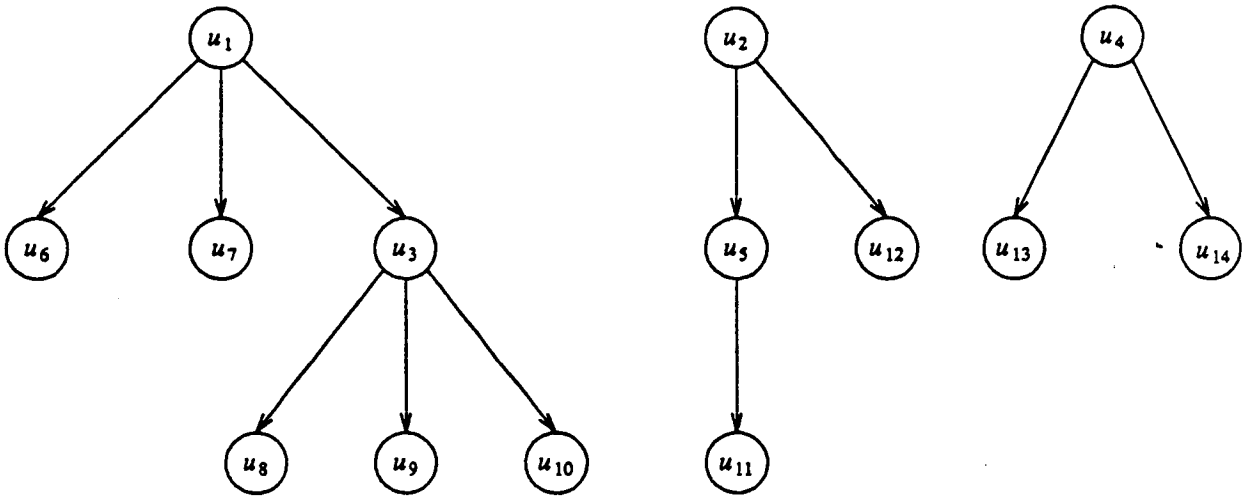


Figure 2 – A forest of embedded trees that is not uniform. $T(u_1) \geq_e T(u_2) \geq_e T(u_4)$, but neither of $T(u_2)$ and $T(u_3)$ is embedded in the other.

processors is denoted by $\pi(F)$. The standard relations of stochastic equality and inequality are denoted by the subscript "*st*," e.g. $\leq_{st}$ denotes "stochastically less than or equal to." A policy π_o is said to be optimal in a class $\mathcal{C}$ of forests, if for any policy π, $\pi_o(F) \leq_{st} \pi(F)$ for all $F \in \mathcal{C}$.

Trivially, forest structure is preserved in the remaining graphs at each decision point of a policy π scheduling a forest F. If F is uniform, then by the transitivity of the embedding relation the uniform property is also preserved in the forests remaining at each decision point.

The class of policies of interest here is unrestricted; in particular it contains the preemptive policies. However, there are two useful reductions that can be made. These observations are standard and apply to directed acyclic graphs in general. Formal proofs of the two lemmas below are easily supplied and left to the interested reader.

Lemma 1. *There exists an optimal policy whose decision points occur only at time* 0 *and all but the last job finishing time.*

In other words, preemptions and new job assignments occur only in the initial state and the states resulting from all but the last job completion. This fact results from the memoryless property of the exponential distribution; between job completions the state represented by the remaining graph and the distributions of remaining running times does not change. Hereafter, we confine ourselves to policies having the property given in Lemma 1.

Lemma 2. *An optimal policy never allows a processor to remain idle if there is an available job, i.e., an unassigned unfinished job all of whose predecessors have finished.*

A policy violating Lemma 2 will be called an *idling* policy. The non-idling property of Lemma 2 follows easily from the fact that policies are allowed to preempt jobs at any

time. Lemma 2 can be extended to include general distributions with infinite support. For distributions with finite support, the lemma must be weakened to the assertion that there exists an optimal policy that never allows a processor to remain idle if there is an available job. By convention, we assume that if a policy runs only $k < K$ jobs in the interval between some pair of decision points, then those jobs are run on processors $P_1, \ldots, P_k$.

A recursive computation of expected makespans under a given policy π is easily expressed. Let $R = R(F)$ be the nonempty set of roots in the forest F and let $S = S(\pi, F)$ denote the subset chosen by π to run on $|S| = K \wedge |R| = \min(K, |R|)$ processors. Then if μ is the rate parameter of the given exponential distribution, we have

$$E[\pi(F)] = \frac{1}{\mu(K \wedge |R|)} + \sum_{u \in S} \frac{1}{K \wedge |R|} E[\pi(F - \{u\})],$$

where $\pi(\phi) = 0$. A Bellman equation for an optimal policy π_o can then be written

$$E[\pi_o(F)] = \frac{1}{K \wedge |R|} \left[\frac{1}{\mu} + \min_{\substack{S \subseteq R \\ |S| = K \wedge |R|}} \sum_{u \in S} E[\pi_o(F - \{u\})] \right].$$

These results form the basis of a computer program that was written to evaluate various policies. Examples given later illustrate the application of this program.

A job u in an in-forest has at most one immediate successor; the set of all successors is the set of jobs in the path from u to the root of the tree containing u. An important consequence for in-forest schedules, which is policy independent, is that the number of jobs available for scheduling is a monotone non-increasing function of time for all samples $\{\tau_i;\ 1 \le i \le N\}$. It is easy to see that this property does not carry over to out-forest schedules, except when forests are sets of chains. This suggests that the general out-forest scheduling problem may well be more difficult than the in-forest one.

From a related point of view, one must expect this greater difficulty because depth in out-forests can not occupy the critical role played by level or height in in-forests. For example, consider the forest in Fig. 3, and with $K=2$, consider the greatest-depth-first policy corresponding to HLF for in-forests. As k becomes large compared to l, it becomes more and more important to schedule job u_3 at the outset, even though its depth is less than that of u_1 and u_2. In general then, both the size $z(u)$ and depth $d(u)$ influence scheduling decisions concerning job u. This issue is discussed further in the next section.

3. Greedy Policies for $K=2$

Two obvious greedy policies, denoted γ_d and γ_s, are as follows: γ_d schedules jobs greatest-depth-first with ties resolved in favor of roots of larger trees, and γ_s schedules jobs largest-tree-first with ties resolved in favor of deeper roots. Ties remaining after both criteria have been applied are broken in favor of lower indexed jobs in both γ_d and γ_s. Figure 3 with $(k, l) = (3, 3)$ gives a counterexample to the optimality of γ_d, and Fig. 4 shows a similar counterexample to the optimality of γ_s. Moreover, Fig. 3 with $(k, l) = (3, 4)$ supplies an example where neither γ_d nor γ_s is optimal. For, it is easily verified numerically that an optimal policy must begin with a greatest-depth-first decision; then, with positive probability, an optimal policy will reach the state in Fig. 3 with $(k, l) = (3, 3)$, wherein a largest-tree-first decision is optimal.

It is difficult to see how to balance structural parameters like depth and size in formulating optimal scheduling decisions. This situation motivates the definition of uniform forests, where depth and size orderings are always the same; i.e., if u and v are jobs in a uniform forest, then either $d(u) \geq d(v)$ and $z(u) \geq z(v)$, or $d(u) \leq d(v)$ and $z(u) \leq z(v)$ throughout that part of a schedule where u and v are still in the forest which remains to be scheduled. Then γ_d and γ_s become the same policy, which we denote

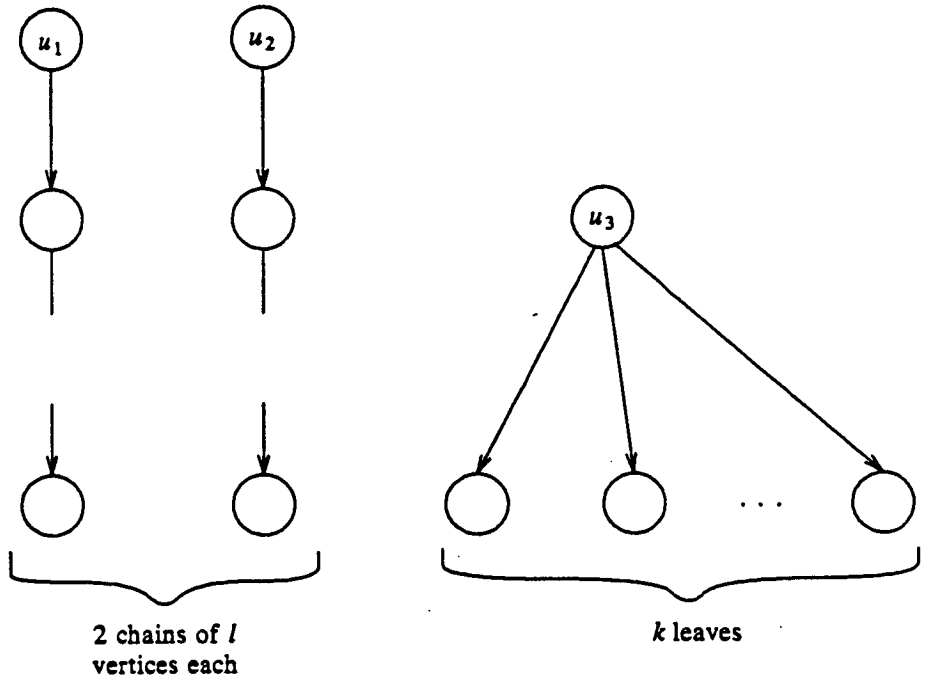


Figure 3 – Counterexample to depth-first

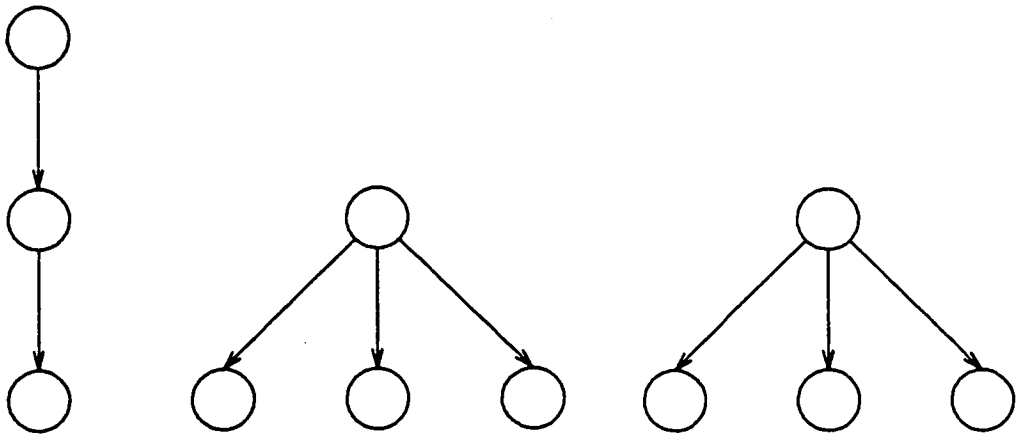


Figure 4 – Counterexample to largest-first

simply by γ when dealing with uniform forests. The theorem below shows that γ is optimal in the class of uniform forests, when $K=2$. The proof uses a coupling argument based on the following well-known result.

Lemma 3 (Strassen [7]). *Two random variables X and Y satisfy $X \leq_{st} Y$ if and only if there exist two random variables $\hat{X}$ and $\hat{Y}$ defined on a common probability space such that $X =_{st} \hat{X}$, $Y =_{st} \hat{Y}$, and $\hat{X} \leq \hat{Y}$ almost surely* (a.s.).

Theorem 1. *Let F be a uniform forest and let π be an arbitrary scheduling policy. Then for $K=2$*

$$\gamma(F) \leq_{st} \pi(F).$$

Proof. First, we explain the coupling argument and the use of Lemma 3. Define $\sigma_1 = \sigma_{10}, \sigma_{11}, \ldots$ and $\sigma_2 = \sigma_{20}, \sigma_{21}, \ldots$ as independent sequences of independent samples from the exponential job running-time distribution. Let forest F be scheduled by γ and π, coupled by the sequences σ_1 and σ_2 as follows. Let t_n, $n \geq 1$, denote the n^{th} epoch when γ or π or both finish a job, and define $t_0 = 0$; i.e. $\{t_n;\ n \geq 0\}$ is the union of the decision points under γ and π, respectively. The jobs, possibly different, assigned or reassigned to P_1 at t_n by γ and π are taken to have the same remaining times $\sigma_{1,n}$. Similarly, any job assigned to P_2 at t_n under γ or π is taken to have remaining time $\sigma_{2,n}$. Within this coupled probability model, let $\hat{\gamma}(F)$ and $\hat{\pi}(F)$ denote the makespans under γ and π, respectively. By the memoryless property of the exponential distribution, it is easy to see that $\hat{\gamma}(F) =_{st} \gamma(F)$ and $\hat{\pi}(F) =_{st} \pi(F)$. To prove the theorem we need only show that, if F is uniform, then

$$\hat{\gamma}(F) \leq \hat{\pi}(F) \quad a.s., \tag{3.1}$$

for by Lemma 3, $\gamma(F) \leq_{st} \pi(F)$ then follows.

Let $F_\gamma(n)$ and $F_\pi(n)$, $n \geq 0$, denote the respective forests remaining at t_n under γ and π in the coupled probability model. It is clear that if $F_\gamma(n)$ and $F_\pi(n)$ are uniform and if $F_\gamma(n) \leq_e F_\pi(n)$, for all $n \geq 0$, then (3.1) holds. The remainder of the proof shows that $F_\gamma(n) \leq_e F_\pi(n)$ for all $n \geq 0$.

The proof is by induction on n. The basis $n=0$ is trivial, since $F_\gamma(0) = F_\pi(0) = F$, so suppose $F_\gamma(n) \leq_e F_\pi(n)$ for some $n \geq 0$. Let u_i, $1 \leq i \leq |F_\gamma(n)|$, and v_i, $1 \leq i \leq |F_\pi(n)|$, denote the jobs of $F_\gamma(n)$ and $F_\pi(n)$, respectively, so that by our convention

(3.2) $T(u_i) \geq_e T(u_{i+1})$, $1 \leq i \leq |F_\gamma(n)|$, $T(v_i) \geq_e T(v_{i+1})$, $1 \leq i \leq |F_\pi(n)|$.

Then by the inductive hypothesis, $F_\gamma(n) \leq_e F_\pi(n)$, we have

$$|F_\gamma(n)| \leq |F_\pi(n)| \tag{3.3}$$

$$T(u_i) \leq_e T(v_i), \quad 1 \leq i \leq |F_\gamma(n)|. \tag{3.4}$$

Let $F_\gamma(n)$ have j roots $u_{r_1}, u_{r_2}, \ldots, u_{r_j}$ and let $F_\pi(n)$ have k roots $v_{s_1}, v_{s_2}, \ldots, v_{s_k}$, with $1 = r_1 < \cdots < r_j$ and $1 = s_1 < \cdots < s_k$. By definition of γ, root $u_{r_1} = u_1$ and, if $j > 1$, root u_{r_2} are assigned by γ at t_n to P_1 and P_2, respectively. If $k > 1$, let v_{s_l} and v_{s_m}, $s_l < s_m$, denote the roots assigned by π at t_n to P_1 and P_2, respectively; if $k = 1$, v_{s_l} denotes the root assigned to P_1.

By the coupling of the running times in the schedules of γ and π the following case analysis suffices, based on the job or jobs that finish at time t_{n+1}.

Case 1. u_1 and v_{s_l} ($s_l \geq 1$) finish at t_{n+1}. Then by (3.3)

$$|F_\gamma(n+1)| = |F_\gamma(n)| - 1 \leq |F_\pi(n)| - 1 = |F_\pi(n+1)|. \tag{3.5}$$

Now u_i and v_i, $s_l + 1 \leq i \leq |F_\gamma(n)|$, are jobs in $F_\gamma(n+1)$ and $F_\pi(n+1)$ respectively, so by (3.4),

(3.6) $$T(u_i) \leq_e T(v_i), \quad s_l + 1 \leq i \leq |F_\gamma(n)|.$$

We have $T(v_i) \leq_e T(v_{i-1})$ by (3.2), so (3.4) shows that

(3.7) $$T(u_i) \leq_e T(v_{i-1}), \quad 2 \leq i \leq s_l.$$

Since $F_\gamma(n+1) = F_\gamma(n) - \{u_1\}$ and $F_\pi(n+1) = F_\pi(n) - \{v_{s_l}\}$, we obtain $F_\gamma(n+1) \leq_e F_\pi(n+1)$ directly from (3.5)-(3.7).

Case 2. $j \geq 2$, $k = 1$, and u_{r_2} ($r_2 \geq 2$) finishes at t_{n+1}. Then by (3.3), (3.4), and the reasoning in Case 1,

$$|F_\gamma(n+1)| = |F_\gamma(n)| - 1 \leq |F_\pi(n)| - 1 < |F_\pi(n+1)|.$$

$$T(u_i) \leq_e T(v_i), \quad 1 \leq i \leq r_2 - 1.$$

and

$$T(u_i) \leq_e T(v_{i-1}), \quad r_2 + 1 \leq i \leq |F_\gamma(n)|.$$

$F_\gamma(n+1) = F_\gamma(n) - \{u_{r_2}\}$ and $F_\pi(n+1) = F_\pi(n)$, so $F_\gamma(n+1) \leq_e F_\pi(n+1)$ follows as in Case 1.

Case 3. $j = 1$, $k \geq 2$, and v_{s_m} ($s_m \geq 2$) finishes at t_{n+1}. In this case, $F_\gamma(n+1) = F_\gamma(n) = T(u_1)$ and by (3.4), $T(u_1) \leq_e T(v_1)$. But since v_{s_m} is the root of a tree other than $T(v_1)$, $T(v_1)$ is in $F_\pi(n+1)$. $F_\gamma(n+1) \leq_e F_\pi(n+1)$ follows at once from

$$F_\gamma(n+1) = T(u_1) \leq_e T(v_1) \leq_e F_\pi(n+1)$$

and the transitivity of the embedding relation.

Case 4. $j \geq 2$, $k \geq 2$, and u_{r_2} and v_{s_m} ($r_2, s_m \geq 2$) finish at t_{n+1}. Trivially, as in Case 1, (3.3) implies

(3.8) $$|F_\gamma(n+1)| \leq |F_\pi(n+1)|.$$

Now suppose $r_2 \leq s_m$. By (3.4) we have

(3.9) $$T(u_i) \leq_e T(v_i), \quad 1 \leq i \leq r_2 - 1, \quad s_m + 1 \leq i \leq |F_\gamma(n)|.$$

By the argument in Case 1,

(3.10) $$T(u_i) \leq_e T(v_{i-1}), \quad r_2 + 1 \leq i \leq s_m,$$

so $F_\gamma(n+1) \leq_e F_\pi(n+1)$ follows from (3.8)-(3.10) and the fact that $F_\gamma(n+1) = F_\gamma(n) - \{u_{r_2}\}$, $F_\pi(n+1) = F_\pi(n) - \{v_{s_m}\}$.

Finally, suppose $r_2 > s_m$. By (3.4),

(3.11)
$$T(u_i) \leq_e T(v_i), \quad 1 \leq i \leq s_m - 1$$
$$T(u_i) \leq_e T(v_i), \quad r_2 + 1 \leq i \leq |F_\gamma(n)|.$$

Next, observe that, since u_1 and u_{r_2} are the lowest indexed roots in $F_\gamma(n)$, $T(u_i)$ is a subtree of $T(u_1)$ for all $1 \leq i \leq r_2 - 1$. Then since $T(v_1)$ and $T(v_{s_m})$ are disjoint trees and since $T(u_1) \leq_e T(v_1)$, we must have

(3.12) $$T(u_i) \leq_e T(v_{i+1}), \quad s_m \leq i \leq r_2 - 1.$$

Then $F_\gamma(n+1) = F_\gamma(n) - \{u_{r_2}\}$ and $F_\pi(n+1) = F_\pi(n) - \{v_{s_m}\}$, along with (3.8), (3.11), and (3.12) allows us to conclude that $F_\gamma(n+1) \leq_e F_\pi(n+1)$.

This completes the induction step and hence the proof that $F_\gamma(n) \leq_e F_\pi(n)$, for all $n \geq 0$. ■

The above proof breaks down in an attempt to extend it to any $K > 2$; it is easily verified that the embedding $F_\gamma(m) \leq_e F_\pi(m)$ is not necessarily preserved at all decision points.

4. The Greedy Policy for K Arbitrary

The optimality of γ can be proved for uniform forests and any $K \geq 2$ if embedding functions are restricted to those mapping roots into roots. When there exists a function f

embedding $T(u)$ in $T(v)$ such that $f(u) = v$, then we say that f is a *root embedding function* and write $T(v) \geq_r T(u)$ or $T(u) \leq_r T(v)$. Extending this concept to forests, we say that a forest F is *root-embedded* if its trees can be ordered by the root embedding relation $\leq_r$. If the set of all subtrees of F can be so ordered then F is said to be *r-uniform*. (Note that the root-embedded forest in Fig. 2 is not r-uniform.) The main result of this section states that, except for the resolution of ties, γ is uniquely optimal in the class of r-uniform forests for all $K \geq 2$. This result follows as an easy corollary to a more general theorem: For any root-embedded forest, the decisions in the initial state must be greedy decisions.

To prove the theorem, we adopt the more convenient, but equivalent probability model of Sec. 3, introduced here for the purpose of coupling the decision points of two different policies scheduling the same forest. Let $\sigma = \sigma_1, \ldots, \sigma_K$ be a set of independent sequences, where $\sigma_j = \sigma_{j0}, \sigma_{j1}, \ldots$ is a sequence of independent samples from the exponential job running-time distribution for each $j = 1, \ldots, K$. If $0 = t_0 < t_1 < \cdots$ denotes the sequence of decision points under some policy π and sample σ, then

$$t_{n+1} = t_n + \min_{1 \leq j \leq k_n} \sigma_{jn}, \qquad n = 0, 1, 2, \ldots,$$

where $k_n \leq K$ is the number of processors assigned by π at t_n. The job finishing at time t_{n+1} is the job, say u, assigned to P_l, where $\sigma_{ln} = \min_{1 \leq j \leq k_n} \sigma_{jn}$; the state at t_{n+1} is then given by $F_\pi(n+1) = F_\pi(n) - \{u\}$. Makespans in the new model are stochastically equal to those in the original model. In addition, the sequence of states $\{F_\pi(n), n \geq 0\}$ at the decision points of π comprise the same stochastic process in the two models, i.e., corresponding joint distributions are equal in the two models. For this reason we simplify the presentation by continuing our previous notation $\pi(F)$, $F_\pi(n)$, etc. in the new model.

Corresponding to the state sequence $\{F_\pi(n), n \geq 0\}$, we introduce the *decision sequence* $\{V_\pi(n), n \geq 0\}$, with $V_\pi(n) = (V_\pi^1(n), V_\pi^2(n), \ldots, V_\pi^K(n))$, where $V_\pi^j(n)$ is the job assigned

by π to processor P_j at time t_n; by convention, $V_\pi^j(n) = 0$ indicates that no job is assigned to P_j at t_n. Additional notation that will be of use is $b_\pi(u)$, which denotes the time when job u begins under policy π, i.e., when u is first assigned to a processor by π, and $c_\pi(u)$, which denotes the completion time of u under π. In all of the above notation, the forest being scheduled will be understood in context.

Theorem 2. *Let F be a root-embedded forest. Then for any $K \geq 2$ the decisions of an optimal policy in the initial state must be greedy decisions.*

Proof. As noted above we adopt the coupling probability model with samples σ. Let F be a root-embedded forest and let π be a policy that makes at least one non-greedy decision in the initial state. We define below a transformation of π to a policy π_* such that $\pi_*(F) = \pi(F)$; in the initial state π_* makes the same decisions as π except that one of π's non-greedy decisions has been replaced by a greedy decision; and π_* is an idling policy. By Lemma 2 π_* can be transformed into a non-idling policy π_{**} such that $\pi_{**}(F) <_{st} \pi_*(F)$ and π_{**} makes the same decisions as π_* in the initial state. Lemma 3 then proves the theorem, since iterating the above argument at most K times shows that we can construct a non-idling policy with a makespan less than $\pi(F)$ and only greedy decisions in the initial state.

The transformation $\pi \to \pi_*$ entails an exchange-type construction. Let one of π's non-greedy decisions at time 0 be the assignment of root u instead of root v, where $T(u) \leq_r T(v)$ and $T(u)$ is smaller than $T(v)$. We construct π_* to be the same as π except for the scheduling of jobs in $T(u)$ and $T(v)$. Policy π_* begins with the same decisions at time 0 as π except that the assignment of u is replaced by the assignment of v. In general, at any decision point t_n, $n \geq 0$, when a job $y \in T(u)$ is scheduled by π, we want π_* to assign the image $f(y) \in T(v)$ under a root embedding function f of $T(u) \leq_r T(v)$. When π assigns a job $y \in T(v)$ and y is not an image of any job in $T(u)$ then we want π_* also to

make the assignment y. But if π assigns $y \in T(v)$ and y *is* an image of some job in $T(u)$, then we want π_* to assign $f^{-1}(y)$. The above prescriptions must be conditioned on the availability under π_* of the appropriate jobs. Formally, the decisions of π_* are constructed from those of π according to the following procedure.

for $m = 0, 1, \ldots, N-1$

for each j *such that* $V^j_\pi(m) \notin T(u) \cup f(T(u))$ *or* $V^j_\pi(m) = 0$

$$V^j_{\pi_*}(m) = V^j_\pi(m). \tag{4.1}$$

for each j *such that* $V^j_\pi(m) \in f(T(u))$

$$V^j_{\pi_*}(m) = \begin{cases} V^j_\pi(m), & \text{if } \pi_* \text{ has not yet finished } V^j_\pi(m) \\ & \text{by } t_m, \text{ i.e. if } V^j_\pi(m) \in F_{\pi_*}(m), \\ f^{-1}(V^j_\pi(m)), & \text{otherwise.} \end{cases} \tag{4.2}$$

for each j *such that* $V^j_\pi(m) \in T(u)$

$$V^j_{\pi_*}(m) = \begin{cases} f(V^j_\pi(m)), & \text{if } f(V^j_\pi(m)) \text{ is available for assignment} \\ & \text{at } t_m \text{ by } \pi_*, \\ V^j_\pi(m), & \text{otherwise.} \end{cases} \tag{4.3}$$

Note that in (4.3) a job x is available if and only if $x \in F_{\pi_*}(m)$, $p(x) \notin F_{\pi_*}(m)$, and x has not already been assigned at t_m according to (4.2). It is clear from (4.1)-(4.3) that π_* has exactly one fewer non-greedy decision at time 0.

Trivially, the decision points of π_* are those of π, viz. $0 = t_0 < t_1 < \cdots < t_{N-1}$, and both π and π_* finish a job at each t_n, $1 \le n \le N$, where t_N is the latest job finishing time under π. Also, it is readily verified from the root-embedding ordering of F that at each t_n both π and π_* assign the same number of jobs. In particular, if $V^j_\pi(m) \in f(T(u))$, then

either $V_\pi^j(m)$ or $f^{-1}(V_\pi^j(m))$ is available under π_* as needed by (4.2); and if $V_\pi^j(m) \in T(u)$, then either $V_\pi^j(m)$ or $f(V_\pi^j(m))$ is available under π_* as needed by (4.3). Then $\pi_*(F) = \pi(F)$ for any sample σ. To make use of this fact we must verify that π_* is a valid policy. For this purpose, it is sufficient to prove that the following claims hold.

Claim 1. *For all* $m = 0, 1, \ldots, N-1$ *and* $x \in F$

$$x \notin F_{\pi_*}(m) \Rightarrow V_{\pi_*}^j(l) \neq x \quad \text{for all} \quad m \leq l \leq N-1, \quad 1 \leq j \leq K.$$

In words, according to any given sample σ, π_* *never assigns a job already finished.*

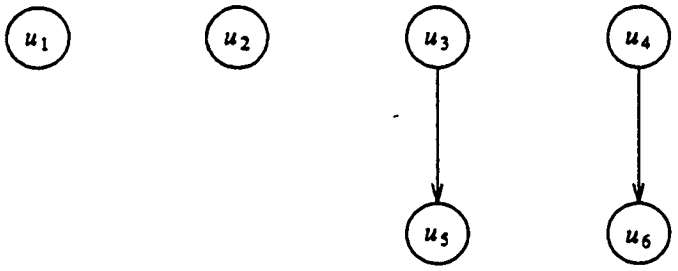
Claim 2. π_* *finishes all jobs; i.e.,* $F_{\pi_*}(N) = \phi$.

Note that Claims 1 and 2 show that $b_{\pi_*}(x)$ and $c_{\pi_*}(x)$ are well defined for all $x \in F$.

Claim 3. π_* *respects precedence constraints: For all* j *and* m, $1 \leq j \leq K$ *and* $0 \leq m \leq N-1$, *if* $V_{\pi_*}^j(m) = x$ *and* x *has at least one predecessor then* $c_{\pi_*}(p(x)) \leq t_m$.

Claim 4. π_* *is non-redundant, i.e.,* π_* *never assigns the the same job to two or more processors at the same time: For all* m, j, k, $0 \leq m \leq N-1$, $1 \leq j < k \leq K$, *if* $V_{\pi_*}^j(m) = V_{\pi_*}^k(m)$ *then both must be* 0, *i.e.* P_j *and* P_k *are not assigned jobs at* t_m.

Given these claims it remains only to observe that π_* is an idling policy at some decision point t_n, $n \geq 1$. Figure 5 shows a simple example illustrating why π_* must be such a policy. As suggested by the figure, it is easy to verify in general that, with positive probability, π_* runs jobs over some interval $[t_n, t_{n+1})$, $n \geq 1$, where there is an idle processor and an available job to run on it. For example, we need only consider those instances σ where the running time of v under π exceeds the sum of the running times under π of all jobs not in $T(v)$. Since $T(u)$ is strictly smaller than $T(v)$, π and π_* must eventually run jobs over some interval in which, for some j,



	π			π_*			
	P_1	P_2	P_3	P_1	P_2	P_3	
t_1:	$\mathbf{u_1}$	u_2	u_3	$\mathbf{u_4}$	u_2	u_3	
t_2:	u_4	u_2	$\mathbf{u_3}$	u_1	u_2	$\mathbf{u_3}$	
t_3:	u_4	$\mathbf{u_2}$	u_5	u_1	$\mathbf{u_2}$	u_5	
t_4:	u_4	0	$\mathbf{u_5}$	u_1	0	$\mathbf{u_5}$	← u_6 available but not scheduled
t_5:	$\mathbf{u_4}$	0	0	$\mathbf{u_1}$	0	0	
t_6:	$\mathbf{u_6}$	0	0	$\mathbf{u_6}$	0	0	

Busy-processor sequences under π and π_*; jobs in boldface finish and create the state transitions.

Figure 5 – Example showing that π_* is an idling policy; $K=3$.

— v is running on P_j under π and u is running on P_j under π_*,

— there is an idle processor under both π and π_*, and

— there is an available root under π_* but not π.

We conclude the proof by establishing Claims 1-4.

Proof of Claim 1. For all jobs in $F - T(u) - f(T(u))$, π_* makes the same assignments as π, by (4.1). Since π is a valid policy, the claim must therefore hold for these jobs under π_*.

Now consider $x \in T(u)$ and bear in mind the property of (4.2) and (4.3) that if π_* assigns $x \in T(u)$ at some decision point, then π must assign either x or $f(x)$ at that decision point. Suppose the claim does not hold for $x \in T(u)$; there exist l and m such that π_* assigns x at t_m but finishes x at $t_l \leq t_m$. Then either x or $f(x)$ must be finished by π at t_l. Assuming that π finishes x at t_l, then π must assign $f(x)$ at time t_m. Then, since π_* assigns x at t_m, we have by (4.2) that π_* must have finished $f(x)$ at some time $t_r \leq t_m$, $r \neq l$. Since π is assumed to finish x at t_l, π must finish $f(x)$ at t_r. This last fact and π's assignment of $f(x)$ at t_m contradict the validity of π.

Analogous reasoning shows that if π is assumed to finish $f(x)$ at t_l and hence assign x at t_m, then we again get a contradiction. Then the claim holds for all $x \in T(u)$. A similar argument proves that the claim holds for all $x \in f(T(u))$. ■

Proof of Claim 2. As noted earlier, π and π_* assign the same number of jobs at each decision point, and at times $t_1, \ldots, t_N$ a job is finished by both π and π_*. Then by Claim 1 the jobs finished by π_* at $t_1, \ldots, t_N$ must all be distinct. Therefore, π_* finishes all jobs. ■

Proof of Claim 3. For all $n=0,1,\ldots,N-1$, π_* and π schedule the jobs in $F_{\pi_*}(n)-T(u)-T(v)$ in the same order. It follows that π_* respects the precedence constraints of all jobs in $F-T(u)-T(v)$.

It is also easy to see that π_* respects the precedence constraints of jobs in $T(v)-f(T(u))$. Indeed, a sample-path analysis according to (4.1)-(4.3) shows that

$$c_{\pi_*}(p(x)) \le c_\pi(x), \qquad x \in f(T(u)) \tag{4.4}$$

$$c_{\pi_*}(p(x)) = c_\pi(x), \qquad x \in T(v) - f(T(u)). \tag{4.5}$$

These two relations yield the desired result for a job x with immediate predecessor $p(x)$,

$$b_{\pi_*}(x) = b_\pi(x) \ge c_\pi(p(x)) \ge c_{\pi_*}(x), \qquad x \in T(v) - f(T(u)).$$

Consider now the jobs in $T(u) \cup f(T(u))$. As a preliminary observation we have, as in the proof of Claim 1, that for any $x \in T(u)$ either

$$c_{\pi_*}(x) \le c_\pi(x), \qquad c_{\pi_*}(f(x)) = c_\pi(f(x))$$

or

$$c_{\pi_*}(x) = c_\pi(f(x)), \qquad c_{\pi_*}(f(x)) = c_\pi(x)$$

holds. Hence,

$$c_{\pi_*}(x) \vee c_{\pi_*}(f(x)) = c_\pi(x) \vee c_\pi(f(x)) \tag{4.6}$$

$$c_{\pi_*}(x) \wedge c_{\pi_*}(f(x)) = c_\pi(x) \wedge c_\pi(f(x)). \tag{4.7}$$

It is also readily verified that similar relations hold for starting times, viz., for $x \in T(u)$,

$$b_{\pi_*}(x) \vee b_{\pi_*}(f(x)) = b_\pi(x) \vee b_\pi(f(x)) \tag{4.8}$$

$$b_{\pi_*}(x) \wedge b_{\pi_*}f(x)) = b_\pi(x) \wedge b_\pi(f(x)). \tag{4.9}$$

We now prove that, for any $x \in T(u) \cup f(T(u))$ with at least one predecessor,

(4.10) $$b_{\pi_*}(x) \geq c_{\pi_*}(p(x)), \quad x \in T(u) \cup f(T(u)),$$

where by convention $c_{\pi_*}(p(x)) = 0$ if x has no predecessors. We consider $x \in f(T(u))$ first. By (4.5)-(4.9), the relation between $b_{\pi_*}(x)$ and $c_{\pi_*}(p(x))$ is determined by four possibilities:

1. $c_{\pi_*}(p(x)) = c_{\pi}(p(x)), \quad b_{\pi_*}(x) = b_{\pi}(x)$
2. $c_{\pi_*}(p(x)) = c_{\pi}(f^{-1}(p(x)), \quad b_{\pi_*}(x) = b_{\pi}(f^{-1}(x))$
3. $c_{\pi_*}(p(x)) = c_{\pi}(f^{-1}(p(x)), \quad b_{\pi_*}(x) = b_{\pi}(x)$
4. $c_{\pi_*}(p(x)) = c_{\pi}(p(x)), \quad b_{\pi_*}(x) = b_{\pi}(f^{-1}(x))$.

In the first case, (4.10) holds trivially since π is a valid policy. In the second case, (4.10) follows from

$$c_{\pi_*}(p(x)) = c_{\pi}(f^{-1}(p(x)) = c_{\pi}(p(f^{-1}(x))) \leq b_{\pi}(f^{-1}(x)) = b_{\pi_*}(x),$$

where the second equality derives from the fact that the root-embedding function f preserves precedence relations. In the third case, we use (4.4) to obtain

$$c_{\pi_*}(p(x)) \leq c_{\pi}(p(x)) \leq b_{\pi}(x) = b_{\pi_*}(x),$$

and in the fourth and last case, $c_{\pi_*}(p(x)) \leq b_{\pi_*}(x)$ follows easily from (4.3). Thus, the claim holds for $x \in f(T(u))$.

The proof of (4.10) for $x \in T(u)$ proceeds along the same lines. We have the following four possibilities:

1. $c_{\pi_*}(p(x)) = c_{\pi}(p(x)), \quad b_{\pi_*}(x) = b_{\pi}(x)$
2. $c_{\pi_*}(p(x)) = c_{\pi}(f(p(x))), \quad b_{\pi_*}(x) = b_{\pi}(f(x))$
3. $c_{\pi_*}(p(x)) = c_{\pi}(f(p(x))), \quad b_{\pi_*}(x) = b_{\pi}(x)$
4. $c_{\pi_*}(p(x)) = c_{\pi}(p(x)), \quad b_{\pi_*}(x) = b_{\pi}(f(x))$.

We obtain (4.10) in the first two cases as before. In the third case, (4.2) yields

$$c_{\pi_*}(p(x)) > c_{\pi_*}(f(p(x))) = c_{\pi_*}(p(f(x)))\,,$$

which implies $b_{\pi_*}(x) \geq b_{\pi_*}(f(x))$. By (4.8) we obtain

$$b_{\pi_*}(x) = b_{\pi_*}(x) \vee b_{\pi_*}(f(x)) \geq b_{\pi}(f(x)) \geq c_{\pi}(p(f(x))) = c_{\pi_*}(p(x))\,.$$

Again using (4.2) for the last case, we obtain

$$b_{\pi_*}(x) \geq c_{\pi_*}(f(x)) > b_{\pi_*}(f(x))\,.$$

It then follows from (4.8) that

$$b_{\pi_*}(x) = b_{\pi_*}(x) \vee b_{\pi_*}(f(x)) \geq b_{\pi}(x) \geq c_{\pi}(p(x)) = c_{\pi_*}(p(x))\,.$$

Thus, (4.10) and hence the claim is proved. ■

Proof of Claim 4. For some $0 \leq m \leq N-1$ and $j \neq k$, suppose that $V^j_{\pi_*}(m) = V^k_{\pi_*}(m) = x$. It is easy to see from (4.1) that a violation of the claim requires $x \in T(u) \cup f(T(u))$. If $x \in T(u)$, then since π is non-redundant, $V^j_{\pi_*}(m)$ and $V^k_{\pi_*}(m)$ must be determined by both (4.2) and (4.3), i.e. $V^j_{\pi}(m) = x$ and $V^k_{\pi}(m) = f(x)$ or $V^j_{\pi}(m) = f(x)$ and $V^k_{\pi}(m) = x$. Then

$$c_{\pi}(x) \wedge c_{\pi}(f(x)) > t_m\,. \tag{4.11}$$

But in this case (4.2) implies $c_{\pi_*}(f(x)) \leq t_m$, which together with (4.11) contradicts (4.7).

Finally, suppose that $x \in f(T(u))$. Again, since π is non-redundant, $V^j_{\pi_*}(m)$ and $V^k_{\pi_*}(m)$ must be determined from both (4.2) and (4.3); one of $V^j_{\pi}(m)$ and $V^k_{\pi}(m)$ must be x and the other must be $f^{-1}(x)$. But under (4.2) and (4.3), x will first be mapped into x and then $f^{-1}(x)$ will be mapped into $f^{-1}(x)$. This contradicts the assumed violation so the claim follows. ■

We have shown that π_* is a valid policy, so the theorem is proved. ■

Since the root-embedding property is preserved at each decision point of a policy scheduling an r-uniform forest, we have the following immediate consequence of Theorem 2.

Corollary 1. *If F is an r-uniform forest, then for all $K \geq 2$,*

$$\gamma(F) \leq_{st} \pi(F)$$

for all policies π. Moreover, the inequality is strict if π ever makes a non-greedy decision, i.e., *if π is optimal, it differs from γ only in the resolution of ties.*

5. Final Remarks

The results of this paper are easily extended to the number-in-system objective function. In particular, under the assumptions of Theorems 1 and 2, γ stochastically minimizes the number of unfinished jobs in the system at any time t. It can also be shown that this optimality is retained in a system with stochastic arrivals of forests that preserve the required uniform property; the arguments needed are just those in [1] trivially adapted to the out-forest model.

The complexity of out-forest stochastic scheduling was the point of departure for this paper. Obviously, the general problem remains open; however, there are other interesting open problems that may be more tractable. For example, it would be useful to know whether results similar to Theorems 1 and 2 are possible within the class of non-preemptive policies. Also, while greedy rules are not always optimal, it may be possible to demonstrate that, within a general probability model of forest structure, their expected performance is close to optimal.

References

[1] F. Baccelli, J. Walrand, "Optimal Processing of a Stream of Trees on Two Parallel Processors," *Systems and Control Letters*, to appear.

[2] J. Bruno, "On Scheduling Tasks with Exponential Service Times and In-Tree Precedence Constraints," *Acta Informatica*, **22** (1985), 139-148.

[3] K. M. Chandy, P. F. Reynolds, "Scheduling Partially Ordered Tasks with Probabilistic Execution Times," *Operating System Review*, **9** (1977), 169-177.

[4] E. Frostig, "A Stochastic Scheduling Problem with Intree Precedence Constraints," *Oper. Res.* **36** (1988), 937-943.

[5] C. H. Papadimitriou, J. N. Tsitsiklis, "On Stochastic Scheduling with In-Tree Precedence Constraints," *SIAM J. Comput.*, **16** (1987), 1-6.

[6] M. Pinedo, G. Weiss, "Scheduling Jobs with Exponentially Distributed Processing Times and Intree Precedence Constraints on Two Parallel Machines," *Oper. Res.*, **33** (1985), 1381-1388.

[7] V. Strassen, "The Existence of Probability Measures with Given Marginals," *Ann. Math. Stat.*, **36** (1965), 423-439.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique